TD de Algorithmique n° 3 : Tri de tableau

20 octobre 2011
(Correction)

I) Le tri par insertion (version récursive)

Le *tri par insertion* (ou *insertion sort*) qu'on a vu dans le TD 1, peut être décrit aussi comme une procédure récursive : pour trier le tableau $A[1,\ldots,n]$ on trie récursivement $A[1,\ldots,n-1]$ et après on insère $A[n]$ dans le tableau trié $A[1,\ldots,n-1]$.

Exercice 1 :

1. Ecrire le psudocode de la procedure TRI-SELECTION2(A) qui trie récursivement le tableau A par insertion.
2. Est ce que la définition par récurrence est vraiment utile dans cet exemple ?

Correction :

1. On écrit d'abord la procédure qui insère le k-ème élément dans le tableau $A[1,\ldots,k-1]$:

   ```
   INSERT(A, k)
   1  cle ← A[k]
   2  i ← k − 1
   3  while (i > 0) et (A[i] > cle)
   4        do A[i + 1] ← A[i]
   5           i ← i − 1
   6  A[i + 1] ← cle
   ```

   Après on écrit la fonction récursive qui trie récursivement le tableau $A[1,\ldots,n]$. Faire attention que, ici, le cas de base est implicite : $n = 0$.

   ```
   TRI-SELECTION-REC(A, n)
   1  if (n > 1)
   2     then TRI-SELECTION-REC(A, n − 1)
   3          INSERT(A, n)
   ```

   Finalement on écrit la procédure principale qui appelle TRI-SELECTION-REC avec les bons arguments.

   ```
   TRI-SELECTION2(A)
   1  TRI-SELECTION-REC(A, LENGTH(A))
   ```

2. Ce n'est pas vraiment utile parce qu'ici la récurrence remplace seulement la boucle while.

II) Le tri par sélection (version récursive)

Le *tri par sélection* (ou *selection sort*) d'un tableau $A[i,\ldots,j]$ à n éléments consiste à déterminer l'élément minimum du tableau, à échanger cet élément avec le premier élément du tableau et à refaire ces opérations sur le tableau $A[i+1,\ldots,j]$, jusqu'à ce que tout le tableau soit trié.

Pierre Boutillier <pierre(dot)boutillier_AT_pps(dot)jussieu(dot)fr>

Exercice 2 :

1. Simuler l'exécution du tri par sélection sur le tableau $A = \langle 94, 73, 58, 62, 85, 93, 65 \rangle$.
2. Ecrire la procédure récursive TRI-SELECTION(A) qui implémente cet algorithme.
3. (facultatif. Ecrire aussi la version itérative)

Correction :

1. A chaque étape, on écrit en rouge l'élément considéré, et en rouge le minimum du tableau considéré (en bleu).
   step 1) $\langle 94, 73, 58, 62, 85, 93, 65 \rangle$.
   step 2) $\langle 58, 73, 94, 62, 85, 93, 65 \rangle$.
   step 3) $\langle 58, 62, 94, 73, 85, 93, 65 \rangle$.
   step 4) $\langle 58, 62, 65, 73, 85, 93, 94 \rangle$.
   step 5) $\langle 58, 62, 65, 73, 85, 93, 94 \rangle$.
   step 6) $\langle 58, 62, 65, 73, 85, 93, 94 \rangle$.
   step 7) $\langle 58, 62, 65, 73, 85, 93, 94 \rangle$.

2. Le pseudo-code de TRI-SEL-REC est le suivant :

   TRI-SEL-REC$(A, start_idx)$
   ```
   1  if (start_idx < LENGTH(A))
   2     then min_idx ← start_idx
   3          for i ← start_idx + 1 to LENGTH(A)
   4              do if (A[i] < A[min_idx])
   5                    then min_idx ← i
   6          echange A[start_idx] ↔ A[min_idx]
   7          TRI-SEL-REC(A, start_idx + 1)
   ```

   La procédure qui "echange $A[i] \leftrightarrow A[k]$" est facile à écrire :

   ECHANGE(A, i, k)
   ```
   1  temp ← A[i]
   2  A[i] ← A[k]
   3  A[k] ← temp
   ```

3. La version iterative est la suivante :

   TRI-SELECTION(A)
   ```
   1  for (i ← 1) to LENGTH(A)
   2       do min_idx ← k such that A[k] = min{A[k] : i ≤ k ≤ LENGTH(A)}
   3          ECHANGE(A, i, min_idx)
   ```

III) Tri rapide

Le *tri rapide* (ou *selection sort*) est une méthode de tri inventée par C.A.R. Hoare en 1962 basée sur la méthode de conception "diviser pour régner". Pour rappel, l'algorithme du tri rapide qui s'applique à un tableau A et deux entiers g, d, s'ecrit comme suit :

```
PARTITION(A, g, d)
 1  pivot ← A[g]
 2  i ← g − 1
 3  j ← d + 1
 4  while true
 5        do repeat j ← j − 1
 6              until (A[j] ≤ pivot)
 7           repeat i ← i + 1
 8              until (A[i] ≥ pivot)
 9           if (i < j)
10              then echange A[i] ↔ A[j]
11              else return j
```

```
TRI-RAPIDE-R(A, g, d)
1  if (g < d)
2     then p ← PARTITION(A, g, d)
3          TRI-RAPIDE-R(A, g, p − 1)
4          TRI-RAPIDE-R(A, p + 1, d)
```

```
TRI-RAPIDE(A)
1  TRI-RAPIDE(A, 1, LENGTH(A))
```

Exercice 3 :

1. Appliquez l'algorithme TRI-RAPIDE sur les tableaux A, B, C suivants :

$$A = \langle 5, 1, 8, 4, 7, 2, 6, 3 \rangle \qquad B = \langle 4, 4, 4, 4, 4, 4, 4, 4 \rangle \qquad C = \langle 48, 17, 59, 11, 99, 76, 21 \rangle$$

2. Donner les arbres d'exécutions.

Correction : Solution. A faire !